AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated hereafter. [Use ~~strikethrough~~ for deleted matter and underlined for added matter.]

1. (Original) A method for optimizing compilation time of a program, the program including at least one block of code, said method comprising steps of:

generating a current hash value for a block of code in the program;

skipping optimization of the block of code if the current hash value equals a prior hash value; and

storing the current hash value in the block of code if the hash value is not equal to the prior hash value for the block of code.

2. (Original) The method of claim 1, wherein the storing a hash value step further comprises:

allocating area for the generated hash value.

3. (Original) The method of claim 1, further comprising:

setting a scope of the least one block of code.

4. (Original) The method of claim 1, wherein the generating a hash value step further comprises:

using a parameter in hashing function to generate the hash value, wherein the parameter is selected from at least one of the group of a code stream, and a data stream.

5. (Original) The method of claim 1, further comprising the step of:

generating a notice when the hash value is not equal to a prior hash value for the block of code.

6. (Original) A system for optimizing compilation time of a program, the program including at least one block of code, comprising:

means for generating a hash value for a block of code in the program;

means for storing the hash value with the block of code if the hash value is not equal to a prior hash value for the block of code; and means for skipping optimization of the block of code if the hash value equals the prior hash value.

7. (Original) The system of claim 6, wherein the storing means further comprises:

means for allocating area for the generated hash value.

8. (Original) The system of claim 6, further comprising:

means for setting a scope of the least one block of code.

9. (Original) The system of claim 6, wherein the hash value is generated using a parameter in the block of code, wherein the parameter is selected from at least one of the group of a code stream, and a data stream.

10. (Original) The system of claim 6, further comprising:

means for generating a notice when the hash value is not equal to a prior hash value for the block of code.

11. (Original) A computer readable medium for optimizing compilation time of a program, the program including at least one block of code, comprising:

logic for generating a hash value for a block of code in the program;

logic for storing the hash value with the block of code if the hash value is not equal to a prior hash value for the block of code; and

logic for skipping optimization of the block of code if the hash value equals the prior hash value.

12. (Original) The computer readable medium of claim 11, wherein said logic for storing a hash value further comprises:

logic for allocating area for the generated hash value..

13. (Original) The computer readable medium of claim 11, further comprising:

logic for setting a scope of the least one block of code.

14. (Original) The computer readable medium of claim 11, wherein the hash value is generated using a parameter in the block of code, wherein the parameter is selected from at least one of the group of a code stream, and a data stream.

15. (Original) The computer readable medium of claim 11, further comprising:

logic for generating a notice when the hash value is not equal to a prior hash value for the block of code.

16. (Currently Amended) A system for optimizing compilation time of a program, comprising:

a compiler that generates ~~the~~ at least one block of code from the program; and

a compilation optimizer, wherein the compilation optimizer further comprises:

logic that generates a hash value for a block of code in the program;

logic that stores the hash value with the block of code if the hash value is not equal to a prior hash value for the block of code; and

logic that skips optimization of the block of code if the hash value equals the prior hash value.

17. (Original) The system of claim 16, wherein the compilation optimizer further comprises:

logic that allocates area for the generated hash value.

18. (Original) The system of claim 16, wherein the compilation optimizer further comprises:

logic that sets a scope of the least one block of code.

19. (Original) The system of claim 16, wherein the hash value is generated using a parameter in the block of code, wherein the parameter is selected from at least one of the group of a code stream, and a data stream.

20. (Original) The system of claim 16, wherein the compilation optimizer further comprises:

logic that generates a notice when the hash value is not equal to a prior hash value for the block of code.

21. (New) The method of claim 1, wherein generating the current hash value further comprises generating the current hash value for a block of intermediate code, and wherein the prior hash value is associated with preexisting object code that corresponds to the block of intermediate code.

22. (New) The method of claim 21, further comprising comparing the current hash value for the block of intermediate code with the prior hash value.

23. (New) The method of claim 22, further comprising:

generating a current object file from the block of intermediate code when the current hash value for the block of intermediate code does not correspond to the prior hash value; and

linking the generated current object file with other blocks of object code associated with the program.

24. (New) The method of claim 22, further comprising:

retrieving the preexisting object code when the current hash value for the block of intermediate code corresponds to the prior hash value; and

linking the preexisting object code with other blocks of object code associated with the program.

25. (New) A method for compiling a program, comprising:

generating a current hash value for a block of intermediate code in the program;

comparing the current hash value with a prior hash value associated with preexisting object code that corresponds to the block of intermediate code;

linking the preexisting object code with other blocks of object code associated with the program when the current hash value corresponds to the prior hash value; and

when the current hash value for the block of intermediate code does not correspond to the prior hash value, generating a current object file from the block of intermediate code when the current hash value for the block of intermediate code does not correspond to the prior hash value; and

linking the generated current object file with other blocks of object code associated with the program.

26. (New) The method of claim 25, further comprising changing a portion of the block of intermediate code in the program such that generating the current hash value and such that comparing the current hash value with the prior hash value validates software maintenance changes.

27. (New) The method of claim 25, further comprising changing a portion of the block of intermediate code in the program such that generating the current hash value and such that comparing the current hash value with the prior hash value identifies changes in the program.